

ELECTRONIC SYSTEMS DIVISION AIR FORCE SYSTEMS COMMAND

HANSCOM AIR FORCE BASE, MASSACHUSETTS

MCI-75-10



August 1974

**THE FEASIBILITY
OF A
SECURE COMMUNICATIONS EXECUTIVE
FOR A
COMMUNICATIONS SYSTEM**

**Approved for
public release;
distribution
unlimited**

**INFORMATION SYSTEMS TECHNOLOGY APPLICATIONS OFFICE
DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS**

20100827148

LEGAL NOTICE

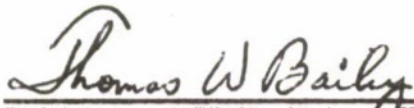
When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

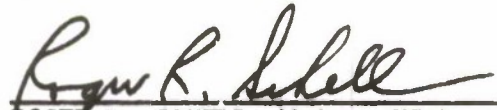
OTHER NOTICES

Do not return this copy. Retain or destroy.

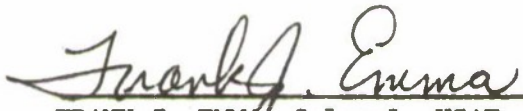
REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.


THOMAS W. BAILEY, Major USAF
Chief, Security Branch
Techniques Engineering Division


ROGER R. SCHELL, Major, USAF
Project Officer
Techniques Engineering Division

FOR THE COMMANDER


FRANK J. EMMA, Colonel, USAF
Director, Information Systems
Technology Applications Office
Deputy for Command & Management Systems

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MCI-75-10	2. GOVT ACCESSION NO. None	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Feasibility of a Secure Communications Executive for a Communications System		5. TYPE OF REPORT & PERIOD COVERED Final Report, July 1974
		6. PERFORMING ORG. REPORT NUMBER MCI-75-10
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Deputy for Command & Management Systems (MCI) Electronic Systems Division (AFSC) L. G. Hanscom AFB, Bedford, MA 01731		10. PROGRAM ELEMENT, PROJECT, TASK APT & WORK UNIT NUMBERS PE 11316F
11. CONTROLLING OFFICE NAME AND ADDRESS See Item 9		12. REPORT DATE 5 August 1974
		13. NUMBER OF PAGES 66
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Security Kernel Secure Communications Executive Internal Access Control *-Property AUTODIN Security Controls		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This document presents the results of the Security Kernel Working Group's study of the feasibility, form and specification inputs required to incorporate the security kernel into a communications processor. The study was performed during July 1974. It was concluded that the kernel can be tailored to provide communications applications support. The limitations and requirements imposed on the system by the use of the kernel are presented. The current DCS AUTODIN security controls and techniques are described in the appendix.		

SECRET

Foreword

The Security Kernel Working Group, whose purpose is outlined in Section 1.1 of this report, conducted its study at the Communications Computer Programming Center (CCPC), Tinker AFB, Oklahoma during the period 1-12 July and at MITRE, Bedford, MA during the period 15-26 July.

LIST OF TABLES

	<u>Page</u>
2.2.2-1 Process Functions	12
4.1.1-1 Process Switches for Typical Input Packet	21
4.1.1-2 Process Switches for Last Packet of Input Message	22
4.1.1-3 Process Switches for Output Packets	23
4.3.2-1 Communications Processor Process Count	27
4.3.2-2 Communication Processor Table Summary	28
4.3.4-1 SCE Space Summary	29
6.3-1 Processor Characteristics Table	43

LIST OF FIGURES

	<u>Page</u>
2.2.1-1 Applications Structure	11
5.1.2-1 SCE Production Schedule	34
5.1.2.5-1 Correspondence Proof of Formal Spec to Model	36

TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ii
LIST OF FIGURES	vi
LIST OF TABLES	vii
1. OVERVIEW	1
1.1 Purpose of the Working Group	1
1.2 Scope of the Study Effort	1
1.3 The Secure Communications Executive	2
1.4 Assumptions Made During the Study Effort	3
1.5 Format of the Paper	4
2. TARGET DESIGN	5
2.1 Secure Communications Executive (SCE)	5
2.1.1 Functional Requirements	5
2.1.2 SCE Commands	5
2.1.2.1 Create Process	5
2.1.2.2 Delete Process	5
2.1.2.3 Block	5
2.1.2.4 Wakeup	6
2.1.2.5 Create Segment	6
2.1.2.6 Delete Segment	6
2.1.2.7 Give Access	6
2.1.2.8 Get Access	6
2.1.2.9 Release Access	6
2.1.2.10 Swap-In	6
2.1.2.11 Swap-Out	6
2.1.2.12 Reconfigure Segments	6
2.1.2.13 Do I/O	7
2.1.3 SCE Data Bases	7
2.1.3.1 Memory Block Table	7
2.1.3.2 Segment Hash Table	7
2.1.3.3 Active Segment Table	7
2.1.3.4 Semaphores (IPC Elements)	7
2.1.3.5 Process Table	8
2.1.3.6 Port Assignment Table	8
2.1.3.7 Process Segment	8
2.1.3.8 Directory	8
2.1.4 Other SCE Considerations	9
2.1.4.1 Utility and Support Functions	9
2.1.4.2 Constraints	9

2.2	Applications Target Design	10
2.2.1	Applications Structure	10
2.2.2	Process Functions	10
3.	SECURITY PROVIDED BY THE SCE	15
3.1	Compromise Prevention	15
3.1.1	Definition of Compromise	15
3.1.2	Compromise Prevention by Means of Internal Access Control	15
3.1.3	Access Domains of a Process and the *-Property	16
3.1.4	Non-SCE "Trusted" Programs	17
3.2	System Integrity	17
3.2.1	Definition	17
3.2.2	The SCE and System Integrity	18
3.2.3	The SCE and Message Integrity	18
3.2.4	The SCE and Functional Integrity	18
3.2.4.1	Functional Integrity of the Applications and Utility/Support Software	19
3.2.4.1.1	Trusted Programs	19
3.2.4.1.2	Other Applications and Utility/ Support Software	19
3.2.4.2	The SCE and Test and Debugging	19
4.	PERFORMANCE	20
4.1	SCE Timing	20
4.1.1	Process Switches	20
4.1.2	Other SCE Functions	24
4.1.3	Total	25
4.2	System Timing	25
4.3	Main Memory Size	26
4.3.1	Resident Code	26
4.3.2	Resident Tables	26
4.3.3	Allocatable Space	28
4.3.4	Summary	29
4.4	Disk Usage	29
5.	SOFTWARE PRODUCTION	31
5.1	SCE Software	31
5.1.1	Production Methodology	31
5.1.2	Software Production Schedule	32
5.1.2.1	Develop Formal Specification Based on Math Model	32

5.1.2.2	Develop High-Level Language Expression Based on Formal Specification	32
5.1.2.3	Translate High-Level Language Expression into Machine Language	33
5.1.2.4	Develop Certified Disassembler	33
5.1.2.5	Prove Correspondence of Formal Specification to Math Model	35
5.1.2.6	Prove Correspondence of Machine Language to Formal Specification	35
5.1.2.6.1	High-Level Language to Formal Specification	35
5.1.2.6.2	Machine Language to High-Level Language	35
5.1.3	Contractor Requirements	37
5.1.4	SCE Load Tape	38
5.1.5	Certification Procedure	38
5.1.6	Modification to SCE Software	38
5.2	Non-SCE Software	39
5.2.1	Multilevel "Trusted" Programs	39
5.2.2	Single-Level Programs	39
6.	SCE REQUIREMENTS	40
6.1	Software Requirements	40
6.1.1	General Software Requirements	40
6.1.2	Software Development Requirements	40
6.1.2.1	High-Level Language Requirements	40
6.1.2.1.1	Mandatory Features	40
6.1.2.1.2	Desirable Features	40
6.1.2.2	Machine Language Requirements	41
6.2	Hardware Requirements	41
6.2.1	Mandatory Features	41
6.2.2	Desirable Features	41
6.3	Existing Hardware	41
7.	CONCLUSIONS OF THE WORKING GROUP	44
APPENDIX		
A.	APPLICABILITY OF AUTODIN SECURITY CONTROLS	A-1

1. Overview

1.1 Purpose of the Working Group

A feasibility study working group was formed at CCPC on 1 July 1974 as a technical working group sponsored by the Deputy for Communications and Navigation, Electronic Systems Division to determine the feasibility, form and specification inputs required to incorporate the kernel into a communications processor (CP). This document represents the results of that effort.

1.2 Scope of the Study Effort

The study effort addressed only the technical aspects of providing security for a CP and avoided assessing a security kernel approach with regard to any cost-related or management-related criteria. To this end, the working group directed its effort toward determining the technical feasibility of a kernel approach to CP security using as a guideline several basic questions.

- (1) Is there hardware to implement the kernel?
- (2) What hardware modifications would be necessary to implement the kernel?
- (3) What coding limitations/bonuses would be involved with implementing the kernel?
- (4) What does the kernel do to throughput?
- (5) What does the kernel do to executive functions?
- (6) Will the kernel adequately protect against compromise?
- (7) What does the kernel do for integrity?

It was felt that the study required to provide the answers to those questions would aid the working group in developing sound conclusions as to the technical feasibility of a kernel approach to CP security.

The effort required to answer these questions and ultimately to develop the conclusions of the study involved a process characterized by the following steps:

- (1) The kernel concept as employed in the PDP 11/45 was discussed followed by discussions of a typical message

switching system.

(2) A trial design which synthesized the two concepts was then developed.

(3) The critical aspects (e.g., timing, sizing, production, and verification methods, hardware availability) of the synthesized approach were investigated.

(4) Answers to the guideline questions were formulated based on the findings of the various aspects of the study, assumptions that had to be made in the course of the study (refer to Section 1.4), and some tailoring of the original kernel concept that was necessary during the course of the study (refer to Section 2).

(5) A set of conclusions was then drawn based upon the answers to the guideline questions and the criteria stated in 1.2 (3).

1.3 The Secure Communications Executive (SCE)

It was found that a significant degree of tailoring of the original security kernel concept (1) was necessary during the course of the feasibility study. As detailed in Section 2 of this report, the functions of the security kernel have been expanded from those which solely perform security management to those which provide not only this feature but also some of those functions performed by an executive in a communications processor. This has resulted in the security kernel taking on a much broader role in communications processing activities.

It is felt that the term "security kernel", as originally conceived, is no longer applicable to the concept of providing security for CP because of the expanded role developed during the study effort for the software module intended to provide the security. The software module has taken on the characteristics of an executive which also performs security management for a communication processor, and as such will henceforth be termed a "Secure Communications Executive" (SCE).

(1) ESD-TR-73-294, Schiller, L., "Design of a Security Kernel for the PDP 11/45", 30 Jun 1973.

1.4 Assumptions Made During the Study Effort

During the course of the feasibility study, it became necessary to make several assumptions to:

(1) Reasonably limit the scope of an assessment to within the time and manpower constraints of the working group.

(2) Allow a calculation or series of calculations to be performed.

(3) Provide a baseline from which assessments or conclusions could be made.

The assumptions made during the feasibility study are listed.

(1) A message consists of 4 packets. Each packet consists of 256 characters (bytes).

(2) The communications system will protect information along 8 security levels, as per AUTODIN.

(3) CPs will handle 6 levels of classified information (exceptions: R (NATO) and M (SI/SAO) traffic).

(4) The SCE is a software device and cannot directly address the hardware aspect of the system integrity question. Integrity as addressed within this paper is narrowed to include only that of the processor software itself.

(5) Code and format conversion must be done by the system.

(6) The hardware as addressed within the paper operates correctly.

(7) For the purposes of timing and sizing calculations, a message passing through the CP will undergo no exception processing.

(8) The MITRE mathematical model of the SCE is appropriate for a CP. (1)

(1) ESD-TR-73-278, Vol I-III, Bell, D. and LaPadula, L., "Secure Computer Systems".

1.5 Format of the Paper

This paper details the findings of the month-long feasibility study effort. Section 2 describes the design features of the SCE with regard to structure and functions performed. Section 3 addresses the means in which the SCE would provide for security and the extent to which it would help maintain system integrity. Section 4 presents calculations of timing and sizing estimates that would be characteristic of a communications processor incorporating an SCE. Section 5 presents the production and verification procedures required to develop the SCE and allow for its ultimate certification. This section also presents a schedule for completing the production and verification procedures for the SCE and briefly addresses the methods of producing and verifying the applications software. Section 6 provides a list of hardware and software requirements for implementing the SCE and matches several eligible machines against the hardware requirements presented. Section 7 presents a list of conclusions to which the working group has come as a result of the study effort.

2. Target Design

2.1 Secure Communications Executive (SCE)

2.1.1 Functional Requirements

The CP SCE will support the execution of a large number of concurrent processes in communication processors. It provides those functions necessary for controlling access to the system's real resources, and provides applications and utility/support software with an efficient means of accessing and managing these resources in terms of logical resources. In doing so, it enforces the security rules for access by processes to resources. The SCE has complete responsibility for management of the hardware elements of the access control mechanism (e.g., description registers) and for access control data bases.

The functions performed by the SCE are as follows:

- (a) Resource control/management
- (b) Interrupt initial processing
- (c) Scheduling
- (d) Storage protection
- (e) Multiprogramming
- (f) Interface control
- (g) Security control

The storage protection function is distinguished from security control, since it is in large part an integrity function.

2.1.2 SCE Commands

2.1.2.1 Create Process

This command will allow a process to create a separate process, having a pre-defined function and clearance.

2.1.2.2 Delete Process

A process may delete itself, or a process it created, from the set operating in the system.

2.1.2.3 Block

In a manner analogous to that used for the P synchronizing primitive, a block command (directed to a specific semaphore) will unbind the calling process from

the physical processor and bind to the processor the highest priority "ready" process.

2.1.2.4 Wakeup

The wakeup command performs a function comparable to that of the V synchronizing primitive, with the added parameters of a priority (for the awakened process) and a 2-word message (to be made available to the awakened process).

2.1.2.5 Create Segment

This command creates a logical segment for subsequent inclusion in the address space of a process.

2.1.2.6 Delete Segment

Removes a previously created logical segment from the set available for inclusion in any process address space.

2.1.2.7 Give Access

Adds a specified process to the access control list of a segment, in a specified mode.

2.1.2.8 Get Access

Associates a hardware descriptor register with a specified segment, if access control rules allow.

2.1.2.9 Release Access

Unbinds a descriptor register from its currently associated segment.

2.1.2.10 Swap-In

Insures that the specified segment is physically located in main memory.

2.1.2.11 Swap-Out

Allows the specified segment to be moved to secondary storage.

2.1.2.12 Reconfigure Segments

Performs, under strict control, the updating of SCE data bases specifying the variable security-related

characteristics of all I/O interfaces.

2.1.2.13 Do I/O

Provides a generalized function through which the SCE can exercise the required degree of control over I/O operations, including the initiation of data transfer to or from specified lines, the reading of the real-time clock and access to device status.

2.1.3 SCE Data Bases

2.1.3.1 Memory Block Table

Flags (allocated, concatenated, free, reserved)	2 bits
Chain or Active Segment Table Entry (ASTE)	
Number	14 bits
Size	1 byte

Total 3 bytes per entry

2.1.3.2 Segment Hash Table

Total 2 bytes per entry

2.1.3.3 Active Segment Table (AST)

Type, status, changed, aged, classification	1 byte
Size	1 byte
Disk address	2 bytes
Chain (for hash table)	2 bytes
Address	2 bytes
Descriptor count, age chain	2 bytes
Access Control List (ACL) head	4 bytes
Category	2 bytes

Total 16 bytes/AST entry, plus space for ACL Entry Pool,
2 bytes/element

2.1.3.4 Semaphores (IPC Elements)

Process pointer	1 byte
Priority	1 byte
Link	1 byte
Message	4 bytes

Total 7 bytes/semaphore

2.1.3.5 Process Table

SCE register storage (highly machine-dependent)	18 bytes
Flags (blocked, ready, inactive) and line	1 byte
Class	1 byte
Category	2 bytes
Process Segment ASTE Number	2 bytes
Semaphore queue head	1 byte

Total 25 bytes/process

2.1.3.6 Port Assignment Table

Address	1 byte
Port ID	1 byte
Classification	1 byte
Category	2 bytes
Function	2 bytes
Process ID	2 bytes

Total 9 bytes/port

2.1.3.7 Process Segment

Descriptor register storage	32 bytes
Process ID	2 bytes
Process number	2 bytes
Classification	1 byte
Category	2 bytes
Segment Descriptor Numbers	64 bytes
SCE Stack	as required

Total 103 bytes plus SCE stack as required, not required to be core-resident.

2.1.3.8 Directory

Type, status, classification	1 byte
ACL Head	1 byte
Category	2 bytes
Disk address	2 bytes
Size	2 bytes

Total 8 bytes/entry, plus space for the ACL elements, 2 bytes/element, not in general core resident.

2.1.4 Other SCE Considerations

2.1.4.1 Utility and Support Functions

Certain functions which might be found in an Executive program are not included in the SCE. These are generally characterized as utility and support functions. Since they are outside the SCE, they are bound by the SCE's security enforcement, but they may where necessary be certified, "trusted" processes.

Functions included in this category are the following:

Trusted Processes

Loader/Initializer
DASD Patch
Program End of Job (EOJ)

Single-level, System High

DASD/Tape Copy and Compare
DASD to Tape to Printer Copy
Memory/Program Dump

Single-level

DASD Dump
DASD/Tape Initializer
Peripheral Error Logger

2.1.4.2 Constraints

The requirement that the SCE provide effective security controls places significant restrictions on the accesses to system resources it allows, as described in Section 3.1.2. As a consequence, a system of applications programs designed to efficiently use the SCE will be influenced by the restrictions placed upon it. In particular, different processes will be required to handle information of different security levels, and communication between processes will be subject to the SCE's security-enforcing rules. Certain objects existing in the system (e.g., queue slots) must be partitioned according to security characteristics, rather than being held in a single homogeneous storage area. These constraints do not limit the functional capability of the overall communications processor; they do affect the manner in which its functions are implemented. More

detailed descriptions are given in the following section.

2.2 Applications Target Design

The applications target design presented directly reflects SCE considerations. The functions, required at a CP, have been organized to better fit an SCE and allow convenient process structuring.

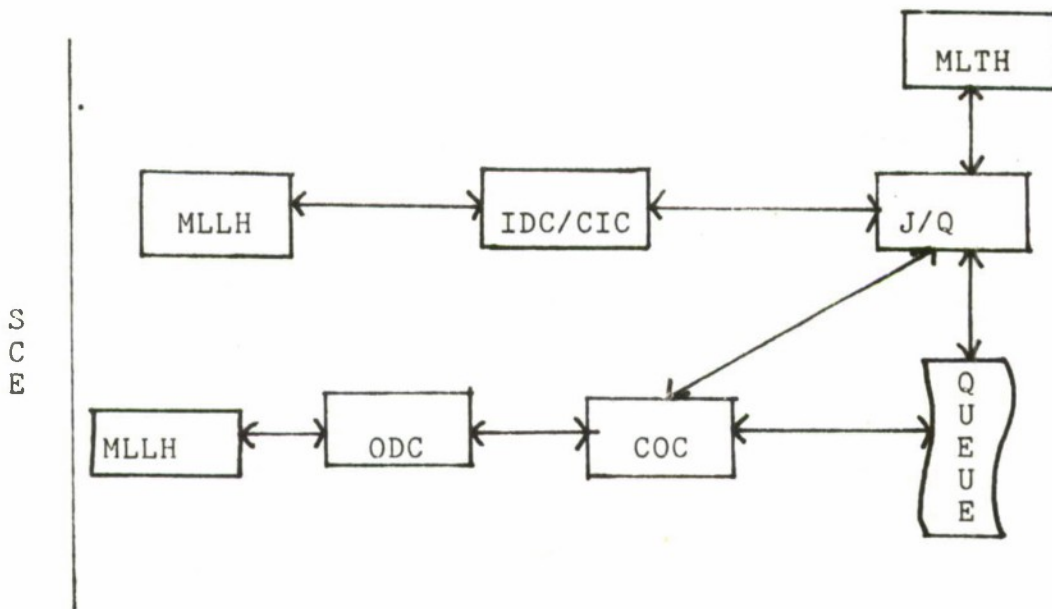
2.2.1 Applications Structure

Figure 2.2.1-1 indicates the processes involved in message switching. This process division would serve well for modular system development.

2.2.2 Process Functions

Table 2.2.2-1 following Figure 2.2.1-1 enumerates the functions associated with each of the processes shown in the Figure for the CP message switching system. These functions are further explained in the applications software specification.

Applications Structure



SCE - SECURITY COMMUNICATIONS EXECUTIVE

MLLH - MULTI-LEVEL LINE HANDLER

IDC - INPUT DEVICE CONTROLLER

ODC - OUTPUT DEVICE CONTROLLER

CIC - COMMON INPUT CONTROLLER

COC - COMMON OUTPUT CONTROLLER

J/Q - JOURNALING/QUEUING

MLTH - MULTI-LEVEL TAPE HANDLER

Figure 2.2.1-1

MLLH Functions:

Line Controller Interface

- Validation
- Timing
- Obtain Buffers and Control Toggle
- Specification of Classification
- Error Handling
- Restart Support
- Line Unloading and Starting Support
- Reconfiguration Considerations
- Trend Analysis Support
- IDC/CIC Interface

IDC/CIC Functions:

- MLLH Interface
- Link Protocol Control
- Line Monitoring
- ADCCP Supervisory Function
- Packet Validation
- Header Validation
- Interface to Message Service Routine
- Message Sequence Number and Date Time Group (DTG)
- Trace Information Adding
- Addressee Summarizing
- Operator Interface
- Code and Format Conversion
- Holdoff of Inputting
- Statistics Collection
- Restart Support
- Device Monitoring
- Error Analysis
- Diagnostics
- Timing
- Establishment of Queue Slot Information
- J/Q Interface

Table 2.2.2-1 Process Functions

J/Q Functions:

- IDC/CIC Interface
- Message Collecting and Timing
- Queue Processing
- Activation of On-Line Retrieval
- Issuing of Partial and Complete Acknowledgement
- Message Deletion After One-Half Hour
- Forced Writes to Journal
- Tape Buffer Management
- Interface to MLTH
- Pushing of Work to Output Device/Line

COC Functions:

- Routing
- ODC Interface
- Operator Interface
- Interface to Message Service Routine
- J/Q Interface for Disposition Reflection
- Swap Out
- Subnet Bit Summarizing

ODC Functions:

- COC Interface
- Swap In
- Code and Format Conversion
- Line Monitoring
- Link Protocol Control
- MLLH Interface
- Acknowledgement Indication or Formulation
- Control of Data Pack Retransmission
- Statistics Collection
- Restart Support
- Device Monitoring
- Error Analysis
- Diagnostics
- Timing

Table 2.2.2-1 Process Functions

Asynchronous Functions:

- Alternate Routing
- Detail Diagnostics
- Auto-Dial Log On
- System Reconfiguration
- Probe Generation
- Site Status Message Generation
- KG34 Initialization
- System Initialization
- System Restart
- USTI
- Console
- Service Message Generation
- Traffic Trace Message Generation
- Statistics Report Formulation
- System Drain
- Intercepting

Table 2.2.2-1 Process Functions (Cont.)

3. Security Provided by the SCE

3.1 Compromise Prevention

3.1.1 Definition of Compromise

A security compromise is defined as one or more of the following conditions taking place:

(1) A terminal or interfaced system receives a message classified to a level higher than the clearance level of the terminal or interfaced system.

(2) A terminal or interfaced system receives a message having a special access category not contained in the set of special access categories authorized for the terminal or interfaced system.

3.1.2 Compromise Prevention by Means of Internal Access Control

A major feature of the SCE is that it effectively prevents compromise that could otherwise occur due to software errors or maliciously inserted software trapdoors. (1) The concepts of a "process" and "access authorization" are necessary to understand the mechanism of the SCE in compromise prevention:

Process - A process in a CP is defined as a program in execution or a virtual machine state. A process is a sequence of processor activity which has a logical identity, a security clearance, access capabilities (implied by its clearance together with access-control-lists associated with information segments), and a well-defined functional responsibility. Each process performs a portion of the total communications processor function.

Access Authorization - A process, P, has access authorization to a segment of information, S, if and only

(1) The SCE does not address compromise that could occur due to hardware failure or deliberately "debugged" hardware. Secure production methods, reliability requirements, and testing must be applied to insure that the hardware performs as specified and does not perform any undocumented operations. The SCE approach relies upon the correctness of the hardware as specified, as does any programmed internal access control mechanism.

if the following are true:

(1) The clearance level (i.e., unclassified, confidential, secret, top secret) of P is greater than or equal to the classification level of S.

(2) The set of special access categories (e.g., SIOP, SI/SAO) attributed to P contains the set of special access categories attributed to S.

(3) P's identification appears on the access-control-list associated with S. The access-control-list entry specifying P will also specify the access mode (e.g., read/write, read/execute) authorized for P to access S.

The function of the SCE as an access control mechanism is to manage the real resource of the bare machine (e.g., the CPU, addressing registers, memory, I/O channel and port interfaces) to allow processes access to only that information for which they have access authorization. The fact that the SCE is verified as functionally correct insures that the access control it provides will be effective. Since all processor activity is directly linked with processes in execution, effectively enforcing every access made by processes according to the access authorization rules insures that the classes of compromise specified in Section 3.1.1 cannot occur.

3.1.3 Access Domains of a Process and the *-Property

The domain of a process refers to the set of segments (data or procedure) currently addressable by the process (i.e., segments which have been made directly accessible in the address space of a process). In the context of the machines under consideration for supporting the SCE, the domain of a process is defined by the address values currently loaded (or designated to be loaded) in the machine's descriptor registers for the process. Only the SCE can alter the domains of processes. The SCE prevents compromise by applying rules which determine whether or not to honor a process' request to add a segment to its domain (with some mode of access). First, the process must have access authorization to the segment. Second, a condition called the *-property must not be violated by inclusion of the desired segment in the domain of the process. The *-property dictates that the domain of a process cannot include read-access to a segment of higher classification than any single segment accessible with write-access in the domain. This property absolutely

prevents a process from downgrading information by reading from a classified segment and writing into a lower classified segment. The machine code executed by the process may contain errors or maliciously planted trap-doors, and yet no program can effect a security compromise due to the restriction on the domains of the processes executing the program.

3.1.4 Non-SCE "Trusted" Programs

The *-property, described above, will not be necessary for the following processes:

1. I/O line-handler processes
2. Tape-handler process
3. System-console process
4. Authentication process
5. System initialization process

These processes will be "multilevel" in that their domains will contain read/write access to segments of various classifications. These multilevel processes will be constrained by the SCE to execute only programs which are proven "trustworthy" (ref. Section 5.2.1) with respect to the criterion of not downgrading information. This is a much simpler proof than verifying the complete functional correctness of these programs, and therefore, the identification of the need for several trustworthy programs is not detrimental to the effective certification of a CP with respect to compromise prevention. Although multilevel processes are not constrained by the *-property, their domains will never include any capabilities for managing the real resources (e.g., in particular, descriptor registers and the SCE's access-control data base) of the system. Those capabilities are reserved exclusively for the SCE.

3.2 System Integrity

3.2.1 Definition

System integrity includes those functions necessary to insure that the information in the system is delivered accurately without distortion or errors introduced during distribution, that the failure of one or more hardware or software elements can be adequately detected and an appropriate restart/recovery procedure initiated and that only authorized terminals or source/destination devices and operators are granted access to the various security levels of the network. System integrity has three

elements: message integrity, functional integrity, and security. Message integrity is concerned primarily with the accuracy and validity of information flowing between information sources and destinations in the system. Functional integrity involves those steps required to detect the failure of hardware or software elements and initiate an appropriate action. Security concerns itself with those actions taken to insure that only authorized terminals or source/destination devices and operators are granted access to the various security levels of the network.

3.2.2 The SCE and System Integrity

All of the elements of system integrity involve both hardware and software. The SCE is a software device and thus cannot directly address the hardware part of the system integrity question. How the SCE addresses system security is answered in Section 3.1. This section (3.2) will discuss what advantages it provides for the elements of message and functional integrity.

3.2.3 The SCE and Message Integrity

The SCE will not prevent improper modification of information by processes having authorized access. The amount of testing necessary to certify that the system possesses message integrity can be reduced by restricting write access to the smallest number of processes possible. The modification of information by any other processes is prevented by the access control features of the SCE. However, proper handling of the data is still dependent upon the correctness of programs executed by processes having authorized access.

3.2.4 The SCE and Functional Integrity

Clearly, system integrity depends upon the correctness of each software module as well as the correctness of each hardware element. The SCE is an access control device which in addition to granting or denying access to system resources to a module also controls the type of access. As such, the SCE has no control over the correctness of any software module. However, since the security enforcing functions of the SCE must be proved correct, their integrity is unquestioned.

3.2.4.1 Functional Integrity of the Applications and Utility/Support Software

3.2.4.1.1 Trusted Programs

Trusted programs will be those modules of software which the executive program (SCE) trusts not to downgrade information. This trust is founded on a formal proof that the module does indeed have this property. As such, trusted programs can be expected to perform security related functions correctly.

3.2.4.1.2 Other Applications and Utility/Support Software

Careful design testing and debugging is the approach to determining applications software correctness. This is because a formal model (analogous to the SCE model) of the functions of the applications software does not exist and would be impractical to synthesize.

3.2.4.2 The SCE and Test and Debugging

The SCE effectively restricts processes to executing programs and accessing data for which the process has access authorization. Verifying the correct execution of a program by a process is simplified since only the authorized domain of the process need be checked to determine the total effect of the computation. Obscure side effects external to the authorized domain are absolutely prevented by the SCE which can also give warning if the process attempts unauthorized access. Moreover, debugging may be simplified since only the processes having authorized access to the errored information need to be checked. The time and effort required to discover and correct an error should be less since fewer programs need be checked.

4. Performance

This section examines the performance of a communications processor using an SCE. The performance estimates are based on the requirements of the maximum configuration, maximum-throughput CP.

Section 4.1 below addresses the fraction of the processor's time that must be allocated to the SCE. Section 4.2 considers overall processor timing including application as well as SCE functions. Section 4.3 presents a computation of SCE main memory requirements and Section 4.4 discusses secondary storage timing.

4.1 SCE Timing

The processor time requirements of the SCE are divided into two major categories. The first includes the processor time allocated to process switches, and the second time allocated to other SCE functions.

The basic throughput requirement for the maximum-throughput CP is 6000 characters per second in plus out. A ten percent over head for packet addresses is required, yielding a total of 6600 characters per second. A typical message is assumed to include four 256-character packets. Dividing the characters per second for the processor by typical message length ($2 \times 256 = 1024$ characters per message) yields 6.45 messages per second in plus out. As the counts of process switches and other SCE operations are on a "per message in" basis, the required figure is input messages per second. Assuming messages-in equal messages-out (single address messages at the CP) the figure above must be divided by two to yield 3.22 messages-in per second.

4.1.1 Process Switches

The count of process switches per message through the CP is based on the target application design of Section 2.2. Tables 4.1.1-1 through 4.1.1-3 summarize the required process switches for a typical four-packet message.

The process switches of Table 4.1.1-1 occur once per packet for each of the first three packets of a message. The switches of Table 4.1.1-2 occur once per message (for the last input packet of the message). The process switches of Table 4.1.1-3 occur for each of the four output packets of the message. Multiplying the counts of

process switches from

Table 4.1.1-1 by 3

Table 4.4.4-2 by 1

Table 4.1.1-3 by 4

and summing yields 84 process switches per message.

Total time per second required for process switches is determined by the time per process switch and the process switches per second.

<u>Switch</u>	<u>Function of process switched to</u>
Cur Proc--MLLH	Create segment for received packet
MLLH--IDC-CIC	Process input packet
IDC-CIC--J/Q	Journal Construct Packet ack
J/Q--ODC	Process Packet ack for output
ODC-MLLH	Output packet ack
MLLH--Cur Proc	Done
Cur Proc--MLLH	Output done
MLLH--ODC	Record output of packet ack
ODC--COC	Ready for next output packet
COC--Cur Proc	Done

Table 4.1.1-1
Process switches for typical input packet

<u>Switch</u>	<u>Function of process switched to</u>
Cur Pro--MLLH	Create segment for received packet
MLLH--IDC-CIC	Process input packet
IDC-CIC--J/Q	Journal; Packet ack
J/Q--MLTH	Write msg to tape
MLTH--Cur Proc	Wait for journal done
Cur Proc--MLTH	Journal done
MLTH--J/Q	Resume journal
J/Q--ODC	Process EOM ack for output
ODC--MLLH	Output EOM ack
MLLH--Cur Proc	Done
Cur Proc--MLLH	Output done
MLLH--ODC	Record output of EOM ack
ODC--COC	Ready for next packet
COC--Cur Proc	Done

Table 4.1.1-2
Process switches for last packet of input message

<u>Switch</u>	<u>Function of process switched to</u>
Cur Proc--MLLH	Ready for more output
MLLH--ODC	Record output done
ODC--COC	Route a packet
COC--ODC	Ready for output
ODC--MLLH	Do output
MLLH--Cur Proc	Done
Cur Proc--MLLH	Packet ack received
MLLH--IDC-CIC	Process Packet ack
IDC-CIC--J/Q	Record packet ack and dismiss packet
J/Q--Cur Proc	

Table 4.1.1-3
Process switches for output packets

Process switches/second =
Messages/second x Process
switches/message

= 3.22 x 84

= 270

Assuming, based on the existing security kernel code for PDP-11/45, a time of 500 us per process switch, a total of 135 ms/second is required for process switches. This figure, indicating 13.5 percent of processor time for process switches, applies for the peak CP load.

4.1.2 Other SCE Functions

The major operations required of the SCE, other than process switches, are those dealing with segment creation, deletion, and swapping to and from main memory. Each four-packet message requires twelve segment creates -- four for received packets, four for transmitted acknowledgement packets, and four for received acknowledgements. Each message also requires eight segment deletes -- for the acknowledgements -- and eight swap-outs and four swap-ins. The excesses of swap-outs over swap-ins and of creates over deletes reflects the fact that messages are allowed to accumulate in the system until disk storage fills.

The total of SCE operations is

12 creates

8 deletes

8 swap-outs

4 swap-ins

or 32 SCE operations per message. Assuming (very conservatively) one millisecond per SCE operation yields:

32 ms/message.

At 3.22 messages/second, the time for other SCE operations is

3.22 x 32 or 103 ms/second

4.1.3 Total

Summing the figures presented above yields

135 ms/second for process switches
103 ms/second for other operations or
238 ms/second for SCE use

This set of figures is based on relatively straightforward exception-free message processing, but on relatively conservative SCE timings. Thus an estimate of 25 to 30 percent of processor time for SCE operations seems conservative.

It should be noted that the estimate above applies to peak message loading and a busy processor (see Section 4.2). If the processor is less busy (less than 6000 characters per second), SCE time requirements will be reduced in proportion.

4.2 System Timing

The system loading of 6000 characters per second for a CP implies a very heavily loaded processor. It is appropriate to consider the processor load on the CP independent of the SCE. A system developed recently by CCPC requires 97 instructions per character in and out to perform its message processing functions. Perhaps 15 percent of the instructions in this system would not apply to a CP, so an estimate of 82 instructions per character seems reasonable. Assuming the same ten percent overhead for packet headers used in Section 4.1 yields:

6600 characters/second

82 instructions/character

or 541000 instructions/second

or 1.84 usec/instruction

The class of minicomputer processors being considered for the CPs are not this fast -- times of 3 usec per instruction or more are more usual.

It should be noted that the problem raised above is independent of the use of a SCE. If a SCE requires more instructions per message than the executive of the CCPC system discussed above, it will aggravate the problem; if less, it will alleviate it. However, it appears that the

basic problem is caused more by the 6000 character per second throughput and the 82 instructions per character than by the use or absence of a SCE. A separate study should address these issues.

4.3 Main Memory Size

This section considers the main memory requirements of the SCE itself. The SCE requires memory space for resident code, for resident tables and data bases, and for allocation on an as-needed basis to data bases such as directories that are not always in main memory. The SCE space requirements have been estimated from the design presented in Section 2.1 and from experience with the existing kernel for the PDP-11/45.

4.3.1 Resident Code

The resident code requirement for a SCE is estimated at 12000 bytes, based on experience with the PDP-11/45 kernel and a fair margin for growth.

4.3.2 Resident Tables

The space requirements for SCE resident tables are dependent on the values of several variables defining the CP configuration and mode of operation. A key factor is the number of processes in the system. Using the process organization shown in Section 2.2.1, 132 processes can be identified as shown in Table 4.3.2-1. The exact number of processes is defined by the number of ports (taken at 38), the number of security levels (taken at 7) and the number of protocols (taken at 3). An additional forty-eight processes are allowed for asynchronous, utility, and support functions, and safety factor. Thus the total process count is 180. There may also be conservatism in the estimate of 132, as it appears that a CP will handle 6 rather than 7 distinct security levels.

1	Input MLLH/Port x 38 ports	38
1	(IDC/CIC)/Protocol/level x 3 Protocols x 7 levels	21
1	(J/Q)/level x 7 levels	7
1	(COC)/level x 7 levels	7
1	(ODC)/level/Protocol x 3 Protocols x 7 levels	21
1	Output MLLH/Port x 38 ports	<u>38</u>

132

Table 4.3.2-1 CP Process Count

The following paragraphs identify specific tables in the SCE, their sizes, and the space they require. Table 4.3.2-2 summarizes this data.

Memory block table: requires 3 bytes per block for each 256-byte block. Assume a 256K memory, for 1024 blocks. Then 3 bytes/block x 1024 blocks = 3072.

Active Segment table: requires 16 bytes per ASTE. The number of ASTE's is estimated at 512, allowing two private segments per process (KS and stack) plus about 150 "general use" shared segments. In addition, 512 2-word "connected process list" entries are allowed in a pool associated with the AST. Thus,
 512 ASTE's x 16 bytes/ASTE = 8192 bytes
 512 CPLEs x 4 bytes/CPLE = 2048 bytes

Memory block table	3072
AST	8192
CPL pool	2048
Hash Table	512
IPC Element Pool	1792
Bit map (Disk)	512
Process table	<u>4500</u>
	20628 bytes

Table 4.3.2-2 CP Table Summary

Hash Table - A hash table, of size about half that of the AST, is used to speed access to ASTE's. Each entry is two bytes, so $2 \text{ bytes/HTE} \times 256 \text{ HTE's} = 512$.

IPC elements - The number of IPC elements need not be much more than the total number of processes. Each element requires 7 bytes, and 256 are allowed so $7 \text{ bytes/IPC} \times 256 \text{ IPC's} = 1792$.

Disk bit map - A 512K byte disk and 256 byte blocks are assumed. At one bit in the map per disk block, the map requires $(1024K/256) \text{ blocks} \times 1/8 \text{ byte/block} = 512 \text{ bytes}$. A disk of this size will allow approximately two minutes of message build-ups before saturating the disk.

Process Table - The process table requires 25 bytes per process. With 180 processes active, the table consumes $25 \text{ bytes/process} \times 180 \text{ processes} = 4500 \text{ bytes}$.

The total requirement for SCE resident tables is thus 20628 bytes.

4.3.3 Allocatable Space

Space is required for allocation to directories and per-process segments as a function of system activity. With three messages per second processed and about 6 processes involved in the handling of each message, it

seems reasonable to allow storage for two seconds worth, or thirty-six sets of per-process segments. Each process requires two per-process segments of 256 bytes (or less).

In addition, a fair number (perhaps fifteen) of directories should be in main memory, at 1024 bytes per directory. This allocation totals:

256 bytes/segment x 72 segments

1024 bytes/directory x 15 directories

or 33792 bytes

This estimate is very crude, so a total of 50K bytes allocatable space is actually allowed.

4.3.4 Summary

Table 4.3.4-1 summarizes the SCE space allocation.

Code	12000 bytes
------	-------------

Tables	20600 bytes
--------	-------------

Allocatable	<u>50000 bytes</u>
-------------	--------------------

	82600 bytes
--	-------------

Table 4.3.4-1 SCE Space Summary

The total allocation to the SCE is thus about 83000 bytes.

4.4 Disk Usage

The throughput requirement for the CP disk is determined by the number of disk reads and writes per message and the number of messages per second. Each message undergoes four swap-ins and eight swap-outs, for a total (1) of twelve disk operations per message. Multiplying this requirement by 3.22 messages per second gives

3.22 messages/second x 12 operations/message

(1)

Note that a swap-in does not require a disk read if a segment's core buffer has not been reallocated.

or 38.6 disk operations/second

This requirement allows 25ms per disk operation. A typical small computer (head per track) disk has a 16 ms access time, so packet processing does not overload the CP disk. If a read after write capability is desired, the disk must have multiple gap heads.

Over and above the requirements for pack input and output are those for program overlays. If each message requires four overlays (swap-ins) the operations per message rise to sixteen, and the total disk usage is

51.5 operations/second

or 19 ms/operation

The latter figure is close to the limit for a small disk. It should be noted that this load, like that on the processor, is a function of the CP loading and not the presence or absence of an SCE.

5. Software Production

5.1 SCE Software

5.1.1 Production Methodology

The SCE software design and implementation will be derived directly from a mathematical model of communications processor security that has been rigorously proven secure. The technique of successive refinement will be used to insure that the ultimate SCE software correctly implements the mathematical model. Four steps of successive refinement that are necessary for the SCE are as follows: Math model; formal specification; high-level language; machine language.

Math Model - MITRE has developed a finite-state mathematical model of computer security which is appropriate for the abstract representation of communications processor security. The model specifies subject and objects, corresponding to processes and segments respectively, access modes, and rules for determining how subjects may access objects. The proof of the model's security (i.e., compromise prevention) insures that any programmed system which accurately corresponds to the model would be incapable of effecting a compromise.

Formal Specification - This representation of the SCE is derived directly from the model and specifies state-variables and operations on them in terms of data structures and algorithms appropriate for implementation on a computer. The formal specification begins to take into account the specific features of the machine (such as the existence of descriptor registers, the I/O architecture, interrupt structure) for which the SCE is to be implemented. The technique used for this level of representation is called a "Parnas Specification." (1)

High Level Language - This representation provides a convenient intermediate step between the formal specification and the machine language for the SCE. The high level language expression is derived directly from the formal specification. Required and desirable features of the high level language are listed in Section 6.

(1) Parnas, D.L., "A Technique for Software Module Specification with Examples," Communications of the ACM, Vol 15, No. 5, May 1972.

Machine Language - The fourth representation of the SCE is the binary code that is loaded into the machine. This code is derived directly from the high-level language through either a compiler program or through manual compilation. An intermediate assembly language version may be useful in this translation from high level language to machine language.

Each successive level of refinement must be proven to correspond correctly and completely to the level of representation from which it was derived. This correspondence will then propagate the security of the mathematical model down to the level of machine language which represents the SCE in an executable form. The techniques for proving the correspondence of each level to its immediate predecessor are described in Section 5.1.2 below.

5.1.2 Software Production Schedule

The production schedule is shown in Figure 5.1.2-1. This schedule assumes that the MITRE finite-state mathematical model will be used. Each task shown in Figure 5.1.2-1 is described below. The SCE production effort will be preceded by a two month lead-time for familiarization with the SCE concepts. The lead-time effort will require four people designated for design/implementation and one person designated for systems engineering/technical support.

5.1.2.1 Develop Formal Specification Based on Math Model

The formal specification will be developed over a four month period with a work level of one or two people for design/implementation and one-half to one for systems engineering/technical direction. The Parnas Specification developed will represent all data structures needed by the SCE to represent machine states and all operations provided by the SCE (i.e., the SCE's primitive functions) for performing state transitions. Applications programmers can begin work at the time the formal specification is complete.

5.1.2.2 Develop High-Level Language Expression Based on Formal Specification

A three month effort with a work level of one person for design/implementation and one-half for systems engineering/technical direction will be applied to the high-level language representation of the SCE.

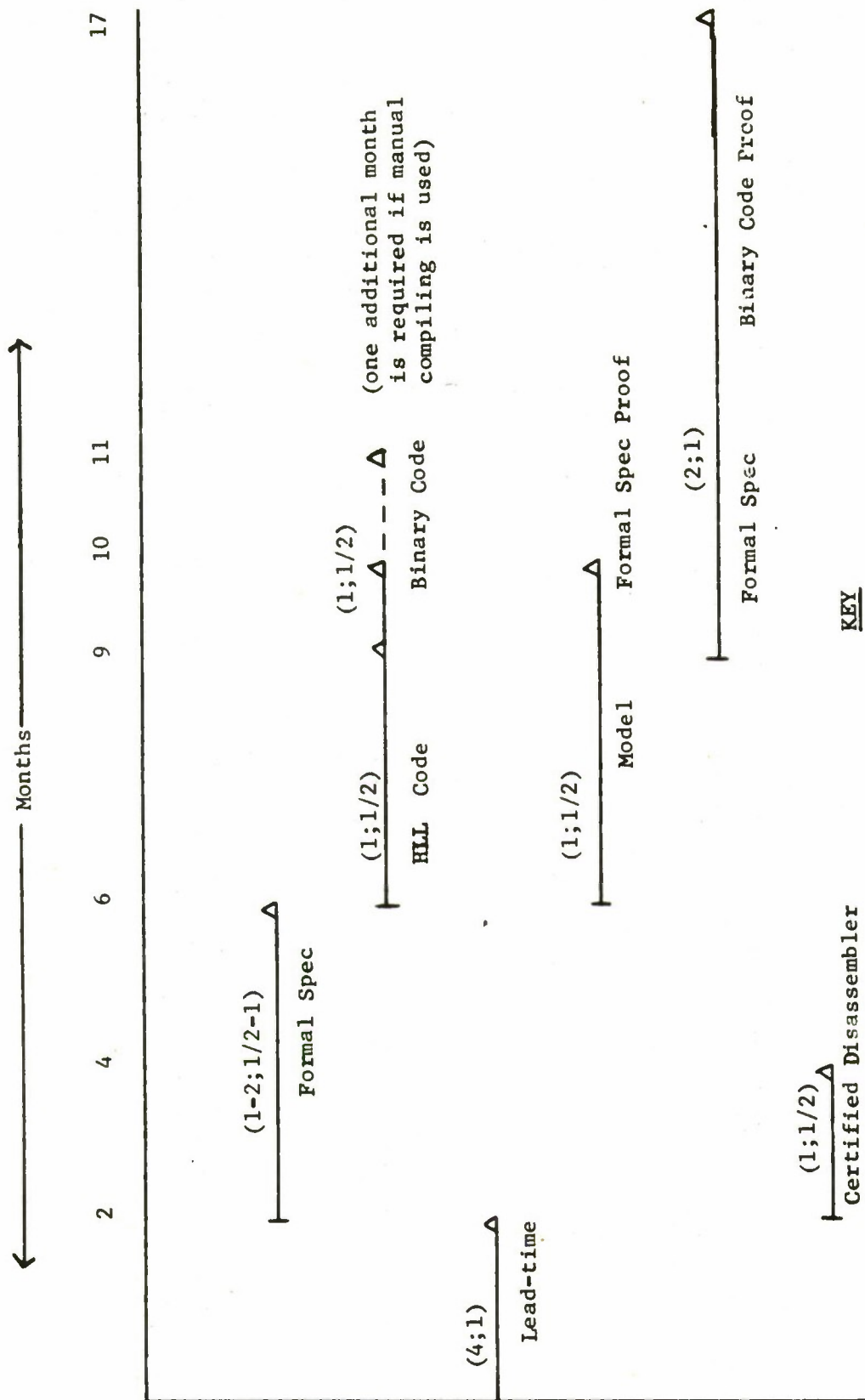
5.1.2.3 Translate High-Level Language Expression into Machine Language

A one month effort with a work level of one person for design/implementation and one-half for systems engineering/technical direction will be applied to this task. This may be accomplished by a compiler program or by manually compiling the high-level language. These two methods are discussed with respect to validation in Section 5.1.2.6. A two month effort will be required if manual compiling is necessary. Applications programmers can begin testing their programs at the time the translation is complete.

5.1.2.4 Develop Certified Disassembler

A two month effort with a work level of one person for design/implementation and one-half for systems engineering/technical direction will be applied to develop a certified disassembler program for use in the correspondence proof of the SCE's machine language expression to the high-level language (See Section 5.1.2.6).

SCE Production Schedule



KEY

Numbers shown as (a;b) are manpower levels.

a - design/implementation

b - systems engineering/technical direction

Figure 5.1.2-1

5.1.2.5 Prove Correspondence of Formal Specification to Math Model

This will require a four month effort with a work level of one person for design/implementation and one-half for systems engineering/technical direction. MITRE has developed a methodology for proving the correspondence of the formal specification to the mathematical model. The states of the mathematical model are first mapped into representations involving the data structures of the Parnas specification level. Then each SCE operation at the Parnas specification level is shown to correspond to the application of a sequence of state-transitions at the math model level for correct correspondence. Figure 5.1.2.5-1 depicts this proof of correspondence.

5.1.2.6 Prove Correspondence of Machine Language to Formal Specification

This will require an eight month effort with a work level of two people for design/implementation and one for systems engineering/technical direction. Two stages are involved: proving the correspondence of the high-level language to the formal specification, and proving the correspondence of the machine language to the high-level language.

5.1.2.6.1 High-Level Language to Formal Specification

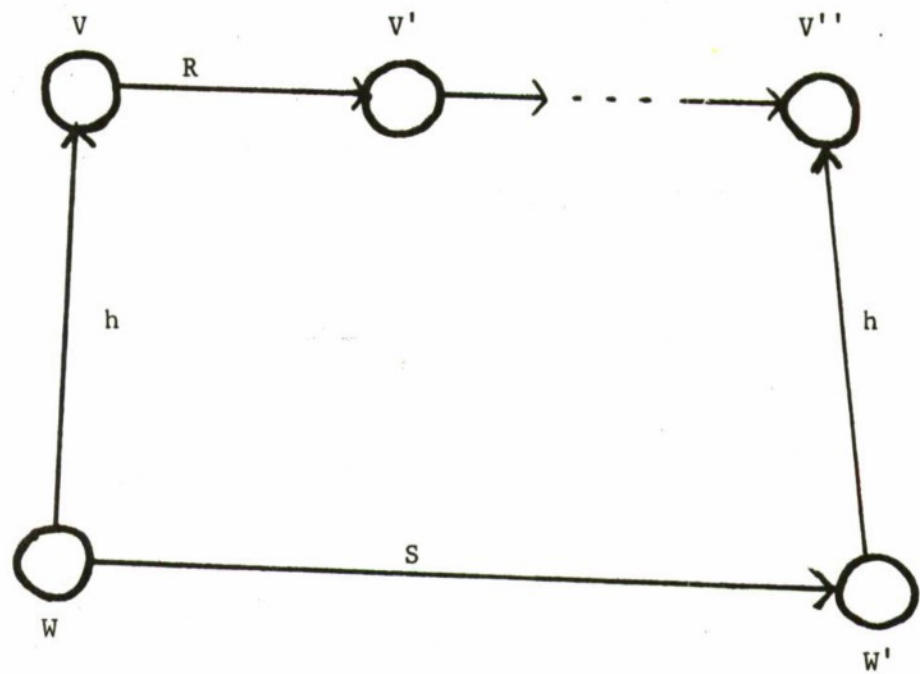
The formal specification will provide a set of assertions for each data structure and SCE operation; these assertions must be correctly preserved in the high level language representation in order to prove the correspondence of the latter to the former. Methods of proof-of-correctness are available which are appropriate for preserving the validity of assertions about programs written in a structured high level language. (1)

5.1.2.6.2 Machine Language to High-Level Language

The SCE object code will be derived directly from the high level language expression of the SCE, based on a statement by statement translation of the SCE's high level language expression. The choice between the following two methods for accomplishing this translation depends upon

(1) Hoare, C. A. R., "An Axiomatic Basis for Computer Programming," Communications of the ACM, Vol 12, No. 10, Oct 1969.

Correspondence Proof of Formal Spec to Model



V - States of the model

W - States (i.e., V-functions) at the Parnas Spec level

h - Mapping between Parnas Spec states and math model states

R - State transformation at the math model level

S - State transformation at the Parnas Spec level

Figure 5.1.2.5-1

whether or not an appropriate compiler exists for the high level language in which the SCE is written:

a. Compiler method: There is no requirement for the compiler used in the translation to be certified correct. The compiler will produce a machine language version of the SCE, and also a mnemonic assembly language listing (complete with symbolic names for registers, storage operands, literals, entry point labels, and a corresponding symbol table) which maps directly to the machine language. The correct correspondence between the assembly language listing and the high-level language expression will be verified manually. A certified disassembler will then be used to disassemble the machine language version of the SCE into mnemonic assembly language. A manual comparison of the compiler's assembly language listing to the disassembler's output will complete the correspondence verification of the machine language to the high level language expression of the SCE.

b. Hand-compile method: In the absence of an appropriate compiler, the high level language expression of the SCE will be manually compiled into an intermediate assembly language or macro-assembly language expression. The correct correspondence between these two expressions will be manually verified. An (uncertified) assembler or macro-assembler will then be used to produce the machine language version of the SCE. Finally, a certified correct disassembler will be used to disassemble the machine language back into assembly language. A manual comparison between the output of the hand compilation and the output of the disassembly will be accomplished to verify the correctness of the machine language version of the SCE.

5.1.3 Contractor Requirements

A contractor will be required to produce all levels of expression of the SCE. The documentation of the correspondence proofs of each level to its predecessor will also be required of the contractor.

The designers and implementors of the SCE will all be cleared to the highest level of information to be processed in the communication system. This is because the correctness verification process takes place concurrently with the work involved in designing and implementing the SCE. Verification of correctness cannot be effectively accomplished unless those individuals who will perform the technical verification are intimately familiar with the SCE. An alternative approach would be

to have system-high cleared personnel closely monitor the efforts of designers and implementors (not cleared system high) throughout the entire software production cycle, as described in Section 5.1.2. The software production facility must be cleared system-high to prevent any unauthorized modification of any media used in the production of the SCE (e.g., design sheets, coding sheets, correctness documentation, cards, tapes, etc.).

5.1.4 SCE Load Tape

The SCE software will be delivered as a single bootstrap tape containing a certified loader routine followed immediately by the SCE machine code and the tables and data bases needed to initialize the SCE in the machine. This tape will be capable of loading itself on the "bare" machine; i.e., it will not require the assistance of any software other than that contained on the tape. The precise machine console and panel switch settings required to initialize the machine for bootstrap using the SCE tape will be specified in the operations documentation.

5.1.5 Certification Procedure

The certifying agencies will conduct a technical assessment of the SCE correctness verification documentation provided by the contractor. For this assessment to be effectively accomplished, qualified personnel must be directly involved with the design and implementation of the SCE throughout the life of the software production effort. ESD can provide technical guidance during the entire SCE assessment period.

5.1.6 Modification to SCE Software

Any modification to the SCE will require re-establishing the correspondence between each modified SCE primitive (at the level modified) and the next higher level of abstraction. Also required is the successive refinement of each modified primitive down to machine language, proving correspondence at each level. This activity will require technical assessment and ultimately recertification.

5.2 Non-SCE Software

5.2.1 Multilevel "Trusted" Programs

These programs will be executed by multilevel processes and, therefore, will be capable of downgrading classified information. These programs must be implemented by two-man teams cleared system-high, since the potential for security compromise exists. Proof-of-correctness techniques with respect to assertions stating that these programs do not downgrade classified material must be applied to these programs. A certification procedure similar to that used for the SCE will be applied to the trusted programs.

5.2.2 Single-Level Programs

These programs are executed by single level processes and, therefore, are incapable of effecting security compromise. However, system integrity is an issue of concern regarding practically all programs. The successful functioning of a common system is dependent on all but a small subset of programs in the system. Certification of programs with respect to system integrity will rely upon software production procedures insuring that software designers and implementors are "non-malicious." Standard test-and-debug methods must be applied to all applications software to achieve a desirable degree of system integrity.

6. SCE Requirements

6.1 Software Requirements

6.1.1 General Software Requirements

- (1) The SCE must be proved correct.
- (2) A class of functions to load the SCE and applications software and to perform the total system initialization must be produced and certified.

6.1.2 Software Development Requirements

As discussed in Section 5, the SCE software should be developed using the technique of successive refinement from mathematical model to machine language code. This development procedure is based on the general software requirements listed above and should use as a guideline the other requirements listed for production of the high-level language and machine language.

6.1.2.1 High-Level Language Requirements

6.1.2.1.1 Mandatory Features

- (1) Structured Programming Language
- (2) ALGOL-like Block Structure
- (3) Data Types: Integer
Character String
Bit String
- (4) Locator of Pointer Variables
- (5) Structured Data Types
- (6) If-Then-Else Statements
- (7) While Loops
- (8) Separately Compiled Subroutines
- (9) Assignment Statements
- (10) Facilities to Handle Interrupts

6.1.2.1.2 Desirable Features

- (1) User-Defined Data Types
- (2) No "GO TO" Statements
- (3) One-Entry Subprograms
- (4) Constraints on Locator or Pointer Variables to Point to Specific Data Types

6.1.2.2 Machine Language Requirements

- (1) Non-Interruptable Test and Set Instructions
- (2) Other Instruction Characteristics as Required in the applications Software Specifications
- (3) Interrupt Control Instructions
- (4) Certified Disassembler

6.2 Hardware Requirements

Based upon the design details of the SCE (refer to Section 2), a list of hardware features necessary to support the software was developed. It should be emphasized that the requirements listed are only those determined to be applicable to the support of SCE software, and do not necessarily include any requirements for supporting the total communications processing activity.

6.2.1 Mandatory Features

- (1) Memory Segmentation
 - a. At least 8 descriptor registers/machine state
 - b. Unique memory access of Read/Execute, Read/Write, No Access
 - c. Minimum Segment Size - 256 Characters
- (2) At Least Two Machine States with Suitable Levels of Privilege

6.2.2 Desirable Features

- (1) Descriptor Base Registers
- (2) I/O Handled as Memory Access - Interrupts Vectored on a Per State Basis
- (3) Unique Memory Access of Read Only
- (4) Demand Paging

6.3 Existing Hardware

The working group listed six machines which were felt to be likely candidates for supporting the SCE as described in Section 2. A study was conducted to determine which of these machines satisfied the hardware requirements listed in Section 6.2. It should be reemphasized that these machines were investigated with regard only to satisfying the requirements for supporting

the SCE and were not analyzed as to their feasibility in the communications processing environment.

It should also be noted that the list includes machines of one particular class, commercial minicomputers. Microprogrammable processors, such as the Burrough's "D" machine, were not investigated in this study, but could prove to satisfy the hardware requirements. Further investigation in this area would be required before any machines in this class could be termed as eligible hardware to support the SCE.

The results of the hardware study are presented in the Processor Characteristics Table which follows.

MANDATORY HARDWARE REQUIREMENTS	COMPUTER					
	DATA CRAFT 6024/4	DATA GENERAL NOVA 840	DEC PDP 11/45	INTERDATA 7/32	MODCOMP IV	PRIME 300
At least two machine states	YES	YES	YES	YES	YES	YES
At least 8 descriptor registers/machine state	YES	YES	YES	YES	YES	YES
Minimum segment size of 256 characters or less	NO	NO	YES	YES	NO	NO
Memory Access: Read/Execute	YES	YES	YES	YES	YES	YES
Memory Access: No Access	YES	YES	YES	YES	YES	YES
Memory Access: Read/Write	YES	YES	YES	YES	YES	YES
DESIRABLE HARDWARE REQUIREMENTS						
Descriptor Base Register	YES	NO	NO	NO	YES	YES
Memory Access: Read Only	YES	NO	YES	YES	YES	NO
Demand Paging	YES	NO	NO	NO	NO	YES
Descriptor-Based I/O	NO	YES	NO	NO	YES	NO
I/O Handled as Memory Access	NO	NO	YES	NO	NO	NO
SUMMARY						
Satisfies All Mandatory Requirements	NO	NO	YES	YES	NO	NO
Satisfies All Desirable Requirements	NO	NO	NO	NO	NO	NO

Table 6.3-1 Processor Characteristics Table

7. Conclusions of the Working Group

Following is the list of conclusions to which the working group has come during its month-long feasibility study effort.

(1) A Secure Communications Executive (SCE) can be realized by tailoring the security kernel to provide communications applications support.

a. A two level machine suffices.

b. Memory segmentation is required and hardware protection support is necessary.

(2) The constraints imposed by SCE design do impact the design and execution of applications programs.

(3) The Secure Communications Executive will use 25-30% of the total CPU time available at maximum throughput. This is believed to be comparable to that used by a conventional executive.

(4) Tailoring the security kernel to a communications function in a two-state machine will increase the amount of code to be certified.

(5) There is no saleable certification scheme on record and certification policy is unclear. The policies and procedures for certifying a computer's security are a parallel development and are not conclusive.

(6) The detailed examination and documentation involved in certification will force a more accurate Secure Communications Executive.

(7) No increase in core requirements will be caused by the SCE design.

(8) With or without the SCE and based on 82 instructions/character, minicomputer processors of the class being considered for the CP cannot support a 6000 character/second throughput.

(9) The use of the SCE does limit the number of machines that can be used.

(10) Of a list of 6 commercial miniprocessors from major manufacturers, two met all the mandatory requirements of Section 6.2 and none met all desirable as

well as mandatory requirements. The one mandatory requirement which was not met by the four other processors was, in every case, the minimum segment size which can be altered by hardware modification.

(11) There is technical risk involved in the SCE proof of correctness. The proof of correctness is still being verified piecemeal in conjunction with other systems, and stating conclusively at this time that no problems will occur would be premature.

(12) The SCE is rigorously verified to enforce access authorization. Therefore, the SCE provides effective compromise prevention.

(13) The SCE will not solve the integrity problem, but does provide an aid in testing system integrity.

APPENDIX A

APPLICABILITY OF AUTODIN
SECURITY CONTROLS

A.1 Overview

A.1.1 Purpose of the Report

This report is a description of current DCS AUTODIN security controls and techniques. The security measures discussed are those which apply to a modern communications system.

A.1.2 Scope of the Report

The AUTODIN techniques are discussed in the light of both hardware and software considerations. Message handling controls are presented in detail, with emphasis on software. It is not the intent of this report to provide a comprehensive threat analysis. Unique system characteristics, such as auto-dial, auto-answer and exception transmission, are not directly addressed. Message switching functions as found at the CP are of prime concern, and packet switching characteristics that exist in the communication system are not addressed.

A.1.3 Background

The seed from which AUTODIN (AUTOMatic DIGital Network) grew was planted by the Air Material Command under the title COMMLOGNET. For this reason, several AUTODIN switches are on or near AFLC bases. The system was originally designed to manage the flow of supply transactions among the Air Material Areas and as such would have been an unsecure system. The basic idea was expanded to become a store and forward message processing system for the DoD and other agencies (e.g., Red Cross, contractors).

Since no proven system for insuring message security in an automated communications system existed in the early '60s, a set of ad hoc rules developed, beginning with the simple concepts of physical security and cleared programmers. These rules have developed to the point where DCA can state that "message security is assured through multiple security checks within the AUTODIN switching centers". DCA manages AUTODIN for the JCS.

A.1.4 Format of the Report

Section 2 describes the software security design with detailed explanations of header and message processing. Section 3 addresses the means by which AUTODIN techniques would provide for compromise protection and message

integrity. Section 4 presents a statement of acceptable performance in light of current system operations. Section 5 presents the software production procedures used in AUTODIN. Section 6 provides a list of hardware requirements and machines which meet these requirements. Section 7 presents conclusions that can be derived from this report and current AUTODIN applications.

A.2 Software Security Design

The security protection features listed below have been incorporated into DCS AUTODIN Switching Center (ASC) software.

A.2.1 Input to the ASC (per-message basis)

A.2.1.1 Each message has a five-character security field consisting of a single character appearing five times. Each character of the field is checked for the presence of a valid code and for perfect agreement with each of the other four characters; an error will result in message rejection.

A.2.1.2 Message security is checked against the line security; (1) if the input line is not cleared for the input message, input message transmission is inhibited.

A.2.1.3 If the addressee is not cleared to receive the message, the associated delivery will not be made and the originator will be notified that message routing is invalid due to security reasons.

A.2.2 Output from the ASC (per-message basis)

Output transmission will be inhibited if either the output line or addressee is not cleared to receive the message.

A.2.3 Input to the ASC (continuous)

On input to the ASC, message data is accumulated into 80-character blocks. Each block is tagged with the proper security code (as specified in the message header) and a sequential number. On all subsequent internal transfers the security code and sequential number of each block are checked; an error or discrepancy will result in supervisory notification and output transmission will be inhibited. The security code attached to each block is transmitted and checked on ASC-ASC transmissions.

A.2.4 Straggler Detection

AUTODIN is programmed to detect input stragglers by comparing message header and trailer station serial number fields for perfect agreement. Any discrepancy will result

(1) These clearance levels are prestored in the ASC.

in message rejection. A straggler is defined as a message, or part of a message, whose address section is controlling the delivery of both. Any straggler is therefore apt to be delivered to the wrong address.

A.2.5 Accountability

A number of ASC internal checks and balances enhance security protection directly and indirectly by providing for the detection of the interlacing of segments of different messages. Some examples are:

A.2.5.1 On-line core resident queue tables contain message block count. On message output actual block count is checked against the queue table block count; a discrepancy results in an appropriate supervisory notification and termination of output transmission.

A.2.5.2 On ASC-ASC trunks the message length or "block count" (number of 80-character blocks) is transmitted in a message control block (MCB). The receiving ASC will verify the sequential block number and MCB block count against the actual count of blocks received; a discrepancy will result in message rejection.

A.2.5.3 On input data messages which contain a block count field, the number of blocks actually received is compared with the block count specified in the message header; a discrepancy will result in message rejection.

A.2.6 Off-line Programs

Off-line support programs which print out data contained on history tapes have been programmed to inhibit or suppress the printing of certain highly classified data unless non-routine extraordinary supervisory intervention is invoked.

A.3 Security Provided by AUTODIN Techniques

A.3.1 Compromise

A.3.1.1 Definition of Compromise

A security compromise is defined as one or more of the following conditions.

1) A terminal or interfaced system receives a message classified to a level higher than the clearance level of the terminal interfaced system.

2) A terminal or interfaced system receives a message having a special access category not contained in the set of special access categories authorized for the terminal or interfaced system.

A.3.1.2 AUTODIN Compromise Prevention Measures

A.3.1.2.1 Message Input

AUTODIN computer checks are made on the redundant security marking characters at the time of message input to assure that:

- 1) they represent a valid security level.
- 2) they are consistent with each other.
- 3) the level they represent is authorized to be transmitted by the transmitting terminal.
- 4) the level they represent is authorized for delivery to the addressee.

A.3.1.2.2 Message Output

On message output from the switch, checks are made to determine that:

- 1) the addressee is authorized to receive the level of classification assigned to the message.
- 2) the output communications channel is cleared to carry the level of classification assigned to the message.

These checks must be repeated at output time even though they were made at the time of message input, in order to protect traffic from changes which may occur in

the communications line status tables. For example, a subscriber's traffic may have been alt-routed to a subscriber who was not cleared to receive the same level of traffic.

A.3.2 Message Integrity

A.3.2.1 Definition

Message integrity includes those functions necessary to insure that the information in the system is delivered accurately without distortion or errors introduced during distribution, that the failure of one or more hardware or software elements can be adequately detected and an appropriate restart/recovery procedure initiated.

A.3.2.2 Failure Recovery Function

Redundant records and storage are maintained on-line at each switch to provide message integrity when minor failures occur in the system. When a catastrophic failure occurs, messages which were in the switch at the time of failure can be retrieved from either of the two history tapes.

A.3.2.3 Service Message Function

If a message does not meet the proper format or is irregular in any way, the system rejects the message and declares an error rather than attempt to continue processing the message. While the required software action is not specified for each message check failure, the following rules generally apply.

- 1) Any time that message check errors are detected at time of message input, the message is either rejected with appropriate control signals to the originator, and/or the switch originates a service message indicating rejection and the reason.

- 2) If a message check error is detected at any other time during processing (i.e., after the switch has accepted the message and is therefore accountable for it), the message is usually "scrubbed" from the system and switch service personnel are responsible for follow-up action to insure that the message is protected.

A.3.2.4 Straggler Function

The AUTODIN computer program checks the incoming message to detect the presence of a straggler. For this purpose the station serial number in the message header is compared for equality to that in the trailer.

A.3.2.5 Communications Line Functions

Parity checks are accomplished on all data transfers within the processor, between processor and peripheral devices, and on communications lines. Parity checking assures integrity of address indicators, precedence, and security markings.

A.3.2.6 Message Block Handling Function

The AUTODIN computer program usually handles messages in segments rather than as whole units. Foolproof accounting procedures are used to prevent inadvertant connection of segments from different messages into one message. To this end, the following checks are used.

- 1) Each segment is sequentially marked on transmission or storage, and the sequential number is checked on receipt or removal from storage.
- 2) Each segment is marked with the classification of the message as indicated by the header whenever transmitted or stored, and consistency of classification marking is checked on receipt or removal from storage.
- 3) The count of message segments on input is compared with the number of message segments being prepared for output.

A.4 Performance

The AUTODIN security techniques exist in several operational systems. Memory requirements, executive overhead, and throughput capacity have been proven to be acceptable in each system.

A.5 Software Production

DCA headquarters maintains complete control of all software. All personnel who assist in the development of program library tapes are cleared for access to the highest classification of traffic processed by the system.

A.5.1 Program Similarity

The on-line programs for each AUTODIN switch are identical and are developed by DCA. The patch areas are also identical and centrally managed. The only variation among these switches occurs in the security authorization data base. This data base is developed and maintained at the switch and contains routing, security, and media information on each subscriber. Although each data base may contain varying numbers of blank entries, all are the same length in order that the on-line programs can be identical.

A.5.2 Program Changes

All program reassemblies are conducted in the secure environment of the test switching centers. (DCA maintains two such test switches - one CONUS configuration and one overseas.) Small changes to the program are implemented through the use of patches. These patches are written, tested and documented at the test switches. The patches are then delivered to the active switches and the source coding changes retained for use in the next reassembly. If the change is to be implemented to comply with a specific operational requirements, the agency concerned (e.g., NSA) will be invited to participate in the formal test and acceptance.

A.5.3 Program Changes (emergency)

Two on-site programmer's (OSP's) are assigned to each switch to provide software maintenance support to the switch supervisor. If a processing error is discovered an OSP will investigate the problem and produce an emergency patch. This patch must be coordinated with the test switch and cannot remain on-line for more than 48 hours. Personnel at the test switch will investigate the problem and generate a properly documented patch.

A.5.4 Program Distribution

The assembly run at the test switch produces an unclassified program library tape (PLT). The PLT is sent

via certified mail to the OSP, who combines it with the security authorization data base to produce the site-unique house operated program (HOP) tape. This HOP is the new on-line system tape and is classified to the highest level of traffic processed by the switch.

A.6 AUTODIN Requirements

A.6.1 Hardware

A.6.1.1 Mandatory Features

A machine with at least two execution states and suitable levels of privilege is required.

A.6.1.2 Desirable Features

- 1) segmented memory
- 2) unique memory access control of read-execute, read-write, no access and read only

A.6.2 Existing Hardware

Of the six commercially available minicomputers surveyed (DATACRAFT 6024/4, DATA GENERAL NOVA 840, DEC PDP 11/45, INTERDATA 7/32, MODCOMP IV and PRIME 3C0), all have all mandatory features and four of the six have all desirable features.

A.7 Conclusions

A.7.1 AUTODIN emphasizes message integrity considerations. Message handling techniques for integrity preservation are well defined, and correct implementation of these techniques assures message integrity.

A.7.2 Use of AUTODIN security techniques in a modern communication system will not impact the program development schedule, since use of these techniques was considered in the projection and specifications.

A.7.3 Application of AUTODIN software security techniques result in acceptable overhead in existing systems, and application of these techniques is not expected to adversely affect throughput.

A.7.4 AUTODIN techniques offer protection against compromise to the degree that software is carefully designed and exhaustively tested.